CONCURRENCY (Part 2)
(EVENT-BASED CONCURRENCY)

I. The Basic Idea

II. An Important API: select()

III. Demo code

IV. Problem and Solution

V. What Is Still Difficult With Event

I. The Basic Idea: An Event Loop

- **Just wait for events and process whenever it comes**

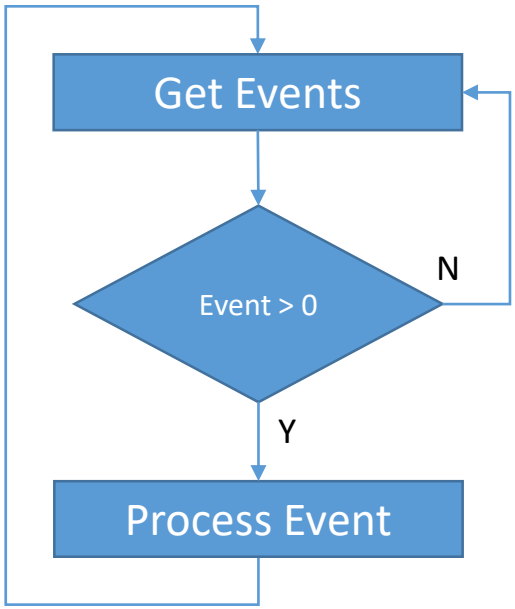
Sample:

```
while (1) {
    events = getEvents();
    for (e in events)
        processEvent(e);
}
```

Advantage:

- No Locks Needed,
- No concurrency bugs

Schedule:



1. The main loop simply waits for something to do (by calling getEvents()

2. If have some event to handler, the next action will be handled. Else, just waiting for the events

3. Depending on what event is received, the program will handle the corresponding tasks

II. An Important API: select()

- ***With that basic event loop in mind, we next must address the question of how to receive events. In most systems, a basic API is available, via either the select() or poll() system calls***

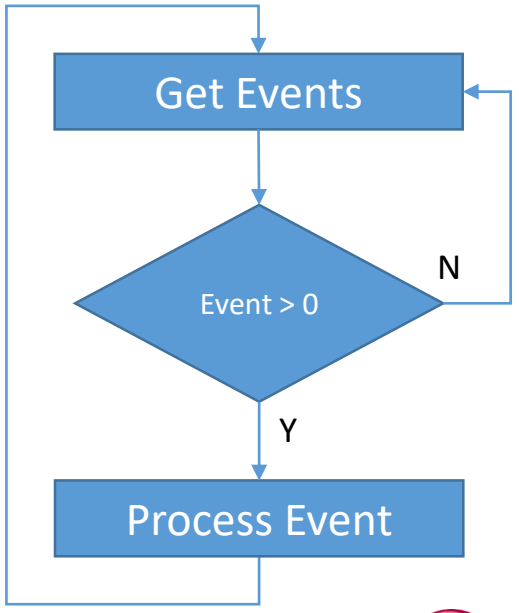
Syntax:

```
int select(int nfds,
           fd_set *restrict readfds,
           fd_set *restrict writefds,
           fd_set *restrict errorfds,
           struct timeval *restrict timeout);
```

Using:

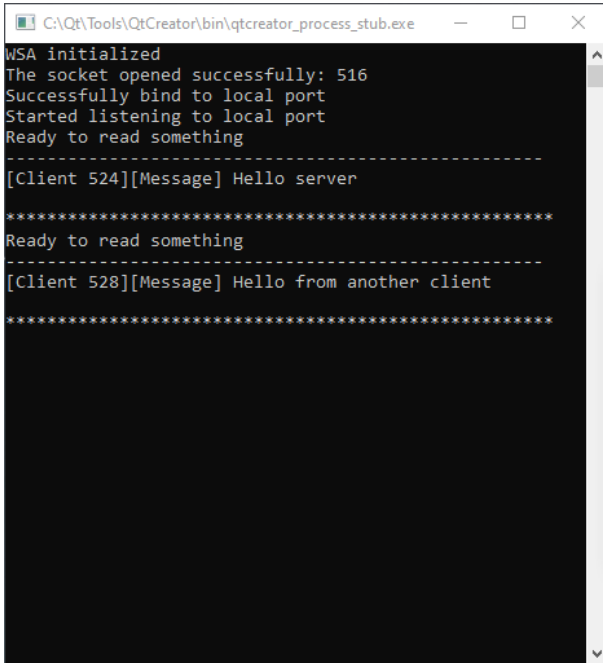
```
while(1){
    FD_ZERO(&fr);
    FD_SET(nSocket, &fr);
    for(int nIndex = 0 ; nIndex < 5 ; nIndex ++){ ...
    //keep waiting for new request and proceed as  per the request
    nRet = select(nMaxFd + 1 , &fr, nullptr, nullptr, nullptr);

    if(nRet > 0){
        //when someone connects to server
        processNewRequest();

    } else if(nRet == 0){
        //nothing to handle
    } else {
        cout <<  "Something Failed" << endl;
    }
}
```

Code demo for using API: select()



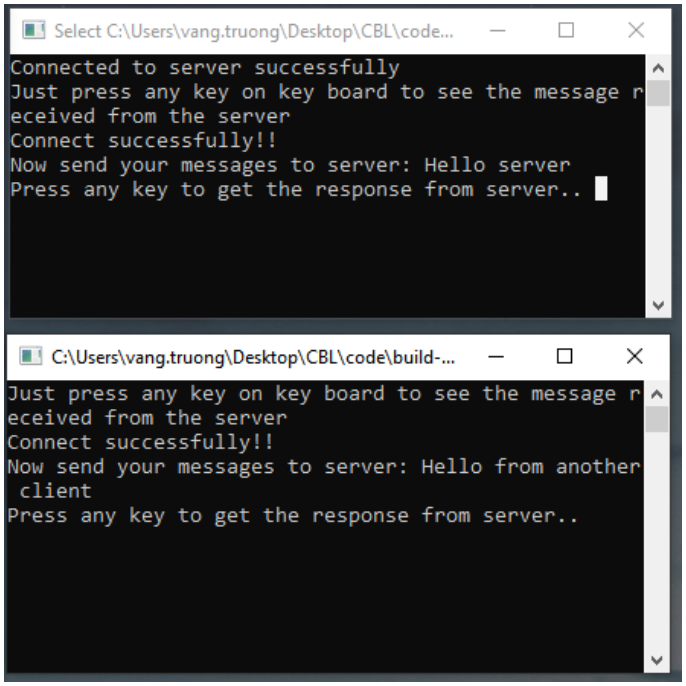
III. Demo code: Client – server communication

- ***This Demo code is designed base on "Event-based Concurrency".***



Server

communicate



Multi clients

- Handshake with clients
- Receive data and send data to client

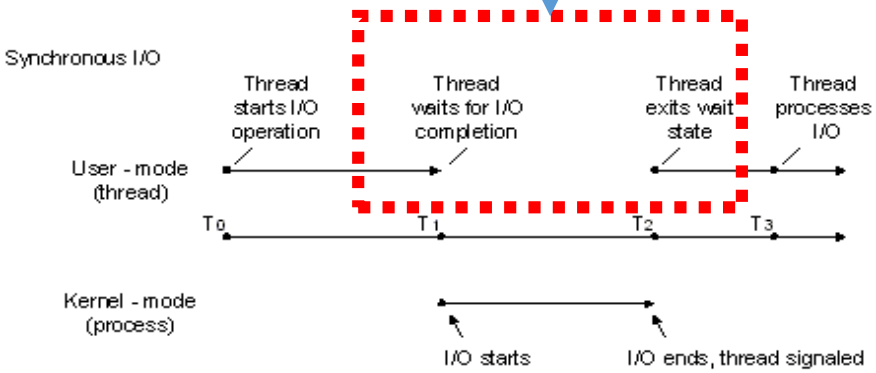
- Connect to server
- Receive data and send data to server

LG
Life's Good

IV. Problem and Solution

Problem: Blocking

When the request needs more time to complete.
In this case, the thread just waiting
→ Waste of resources



File Type	Category
Block Device	Fast
Pipe	Slow
Socket	Slow
Regular	Fast
Directory	Fast

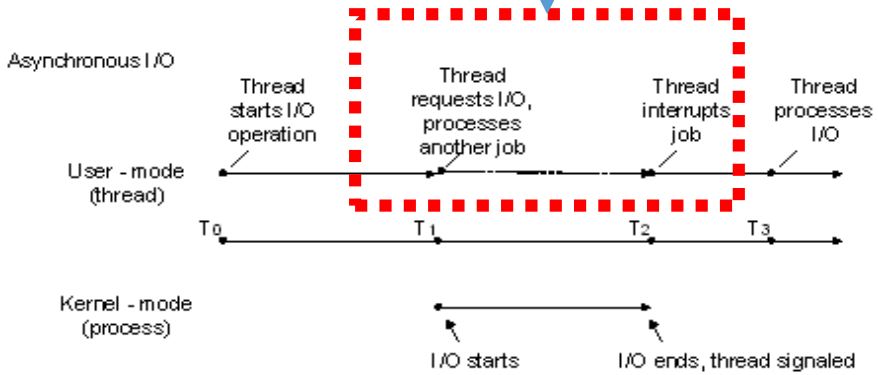
File Type in Unix

LG Life's Good

IV. Problem and Solution

Solution: Asynchronous

When the request needed more time to complete.
In this case, the thread processed another job
→ It look better



API

- int aio_read(struct aiocb *aiocbp);
- int aio_error(const struct aiocb *aiocbp);

API (Qt 6)

https://www.qt.io/blog/asynchronous-apis-in-qt-6

V. What Is Still Difficult With Event

- **CPU have multi threads**
  - For example, when systems moved from a single CPU to multiple CPUs, some of the simplicity of the event-based approach disappeared
  - In order to utilize more than one CPU, the event server has to run multiple event handlers in parallel

- **Does not integrate well with certain kinds of systems**
  - If an event-handler page faults, it will block, and thus the server will not make progress until the page fault completes

- **Flexible**
  - If the code need modify, This model can be more difficult to improve

Thank You

LG
Life's Good